Fruit Quality Classification with Convolutional Neural Networks

Course name: Deep Learning (SoftUni)

Course date: 10.12.2024 - 15.02.2025

C*reator*: Aneta Spasova

Table of Contents

Abstract

Efficient food supply chains and the reduction of food waste play a crucial role in modern societies. A reliable prediction of the quality of fruits could help facilitate automatic harvesting and automatic sorting out of rotten fruits. This paper describes an approach to automatically predict the condition of fruits via the application of Convolutional Neural Networks. For this purpose, two Convolutional Neural Networks were built and benchmarked against a pre-trained VGG16 model. All three models performed well in terms of Accuracy, Recall, Recall and F1-score, which showed that it is possible to predict the quality of fruits reliably. Nevertheless, the Transfer Learning approach with the VGG16 model delivered the best results while at the same time keeping Runtime and Complexity on a low level. Overall, the paper therefore gives cause for optimism that automated fruit recognition can lead to technical innovations in the food supply chain in the near future.

1 Introduction

Food waste is one of the greatest problems and challenges of our time. According to the Food and Agriculture Organization of the United Nations, "28 percent of the world's agricultural area is used annually to produce food that is lost or wasted" (*Food Wastage Footprint*, 2013). One example of these lost or wasted foods is fruits. Fruits have a fast perishability and not consuming them in time leads to food waste. Today 12 percent of the fruits that end up in supermarkets are never sold (*Food Waste, by the Numbers*, 2018). Most of the time fruits are sold in packs which has advantages but also creates two problems. First, a rotten fruit can easily infect other fruits in the same pack. Secondly, if there is only one recognisably rotten fruit, the whole pack will not be bought and as a result also good fruits will be thrown away. Having rotten fruits sorted out by supermarket staff (without the help of machine learning models) would be very subjective and inefficient (Bhargava & Bansal, 2021[1]). Therefore, a sophisticated Fruit Quality Classification system would have benefits. Such a model could also be used in earlier stages of the food supply chain, would save costs and make the entire food production process more efficient.

While previous research has focused mostly on the classification of fruits themselves (Hossain et al., 2019; Y. Zhang et al., 2014[2]), this paper tests the ability of machine learning models to classify the quality of fruits. Three algorithms will be trained using over 24,000 images to detect rotten fruits. Twocustom Convolutional Neural Networks (CNN) are applied and compared to a pre-trained VGG16 model.

1 Fruits and vegetables quality evaluation using computer vision: A review

2 Automatic Fruit Classification Using Deep Learning for Industrial Applications

2 Related Work

Fruit Classification has been the subject of several scientific projects. The idea of Fruit Classification is to feed images of fruits to an algorithm who predicts the kind of fruit that is displayed on a picture. A breakthrough in this area was achieved by Zhang et. al. The team was among the first to apply a CNN in combination with Data Augmentation on Fruit Classification (Y.-D. Zhang et al., 2019[3]). The team created a 13-layer deep CNN. As part of the Preprocessing, images were augmented via image rotation, gamma correction and noise injection (Y.-D. Zhang et al., 2019). An Accuracy of 94.94% was achieved, which was five percentage points higher than previous state-of-the-art approaches.

Kausar et al. (2018)[4] showed that pure Convolutional Neural Networks (PCNN) may have certain advantages over mixed CNN within Fruit Classification. The team concluded that using a GAP layer instead of a fully connected output layer leads to a better performance. Their PCNN with a GAP layer was applied on a Kaggle dataset with 81 different fruits and achieved an Accuracy of 98.88% (Kausar et al., 2018).

As within other areas of deep learning, Transfer Learning is applicable to fruit categorization as well. Siddiqi (2019) showed that Transfer Learning and fine tuning can lead to excellent accuracies in automated fruit image classification. Siddiqi employed Keras' Inception v3 and VGG16 models on a dataset that contained 72 different classes of fruits with 48,249 images in total (Siddiqi, 2019[5]). Overall the fine-tuned VGG16 model achieved the highest Accuracy (99.27%).

An area that is less often investigated is the prediction of fruits in combination with their condition. Dandavate & Patodkar (2020)[6] were among the first to build a classification algorithm that determines not only the fruit, but also its stage (Dandavate & Patodkar, 2020). Images of bananas, papayas, mangos and guavas were therefore classified into 'raw', 'ripe' and 'over-ripe'. The team was able to achieve an Accuracy of 97.74% in only eight epochs and with a validation Recall of 98.33%. Their CNN consisted of eight layers: Two convolutional layers, two pooling layers, one flattening layer and three dense layers (Dandavate & Patodkar, 2020).

This project follows a similar approach to Dandavate & Patodkar, but instead of choosing multiple labels consisting of a combination of fruit and stage, I chose to only predict the quality of the fruit ('Rotten' or 'Fresh'). Focusing on the state of the fruit rather than the combination of type of fruit and state of fruit will be more appropriate for the goal of this project: Identifying rotten fruits to prevent food waste.

3 Image based fruit category classification by 13-layer deep CNN and data augmentation
4 Pure-CNN: A Framework for Fruit Images Classification
5 Effectiveness of Transfer Learning and Fine Tuning in Automated Fruit Image Classification
6 CNN and Data Augmentation Based Fruit Classification Model

3 Conceptual Framework

3.1 Preprocessing Strategy

To achieve a high performance of learning models, first a basic level of quality needs to be ensured across all data points the models are trained and evaluated on. In the context of this project, each data point is represented by an image of a fruit which was either fresh or rotten and an according label. The Preprocessing of these images consisted of three main stages that are described in greater detail in Chapter 4.2. In the first part of the Preprocessing, I filtered the raw data according to several different criteria. After that, the data passed through multiple normalization steps in order to achieve a uniform data format across all instances. Data Augmentation was used to avoid overfitting of the models and increase the number of available training data. The overall Preprocessing process is illustrated in Figure 1.

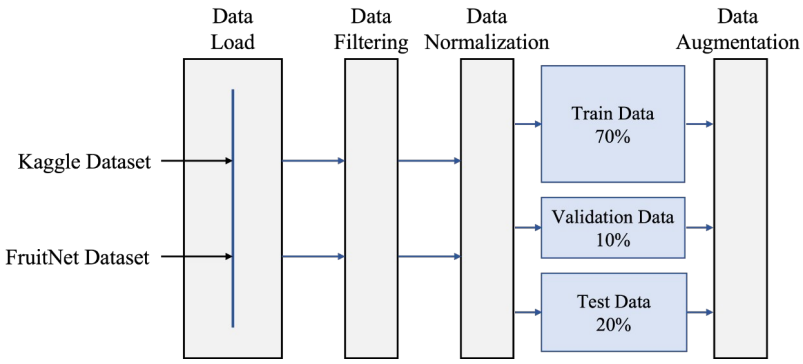


Figure 1: Preprocessing Strategy

3.2 Training Strategy

The goal of the project is to generate a model that predicts whether a fruit is fresh or rotten given an image of the fruit under observation. The project considered three different approaches for classifying the fruit data. All of them fall under the category of CNN. The first two models are self-constructed models of different depth. I will further refer to them as CNN1 for the narrower model and CNN2 for the deeper model. In addition, the training strategy also considers the VGG16 model which leverages pre-trained weights. The pre-trained model should serve as a reference in the evaluation of the self-constructed models. A detailed description of the models is depicted in Chapter 4.3.3. All models will be trained and evaluated on the same data.

4 Methodology

4.1 Dataset Description

I used two public datasets to train and evaluate the different models. The first dataset is from the online platform kaggle.com and contains 2 698 images of fresh and rotten apples, bananas and oranges. This dataset will be referred to as the "Kaggle dataset[7]" in the rest of this paper. Appendix 1 shows a selection of samples from each class and their corresponding labels. When looking at samples from the Kaggle dataset, I saw that the dataset already contained pre-augmented image files. A check of filenames confirmed this assumption. Figure 2 shows one example of an image which existed in six different variants in the Kaggle dataset. To avoid overfitting and for better control over the raw data, augmented versions of images were filtered out from the Kaggle dataset in the Preprocessing (see Chapter 4.2.1).

Figure 2: Sample of Augmented Pictures from Kaggle Dataset of "apples_fresh"

The second dataset was published in the open free repository "Mendeley Data" (Meshram & Patil, 2021[8]). It contains 10 901 images of "fresh" and "rotten" fruits, namely apples, bananas and oranges. For samples, see Appendix 2. This dataset will from now on be referred to as the "FruitNet dataset[9]". A detailed overview of the number of different images of both datasets can be found in Appendix 3. After both datasets were merged, the data was analyzed. First, different images were examined to see whether the data is homogeneous with regards to aspect ratio (height divided by weight) and squared image size. As shown in Figure 3, images for fresh and rotten fruit vary strongly in both dimensions. This also indicates that the data is not biased in this respect.

7 Fruits fresh and rotten for classification and Fruit + quality classifier for NOOBS
8 FruitNet: Indian fruits image dataset with quality for machine learning applications
9 FruitNet dataset

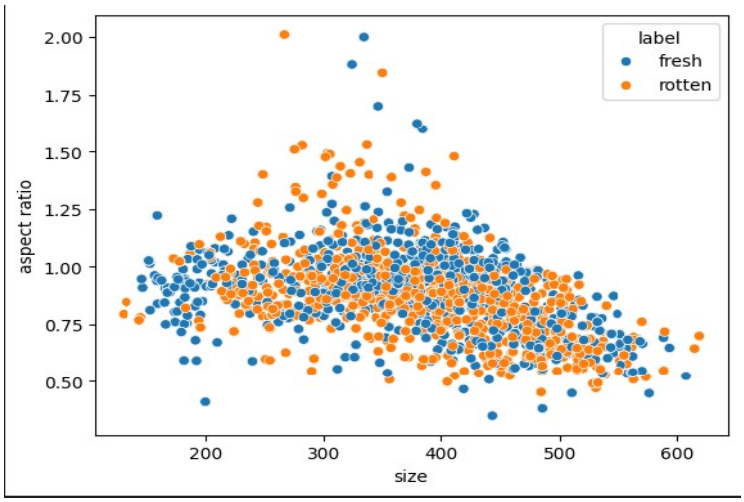


Figure 3: Size and Aspect Ratio

The average RGB values of all images (Figure 4) showed that there are differences between fresh and rotten fruits. However, the deviations were different for all fruit types, i.e. one cannot assume a consistent deviation in one direction.

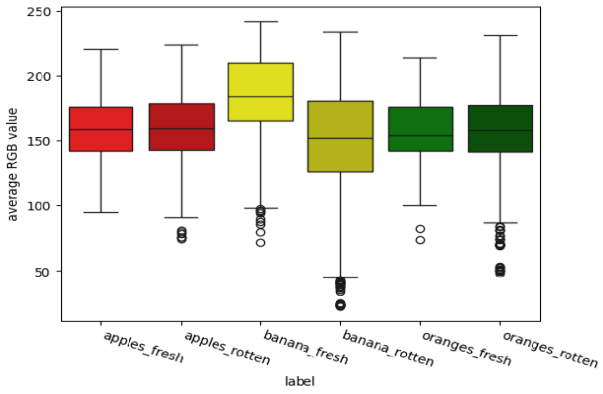


Figure 4: Average RGB values of different fruits and qualities

Finally, the datasets were examined regarding their data balance. A balanced dataset is relevant to avoid misinterpretation of performance metrics such as Accuracy. Figure 5 shows that there are more apples and bananas than oranges.

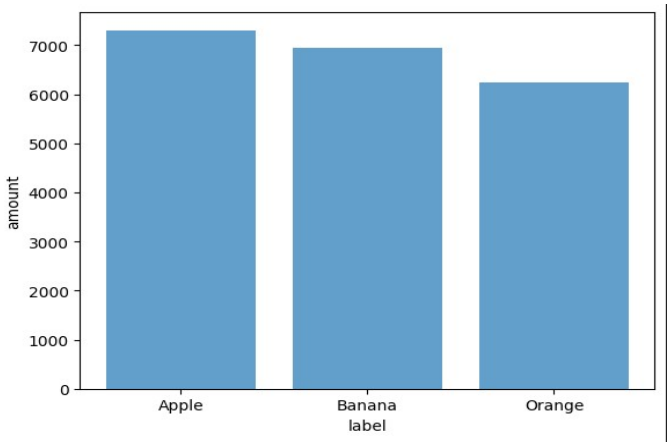


Figure 5: Fruit Balance (Distribution of Fruits)

These models aim to classify the quality of all fruits. I do not want to classify the type of fruit, only the condition of the fruit. The class balance, i.e. the distribution of good and bad fruits, is equal.

4.2 Data Preprocessing

4.2.1 Data Filtering

The first stage of Preprocessing focused on filtering the data. During the exploratory analysis, I discovered several issues in the data that could potentially lower the performance of my models and ultimately result in false predictions. One first step here was to filter out pre-augmented pictures from the Kaggle dataset to stay in control of the augmentation process. Fortunately, all pre-augmented images were named according to the type of augmentation that was applied to them. That way, the filtering operation could be based on whether a certain prefix indicating an augmentation was included in the file name or not. This first filtering step was not applied to the FruitNet dataset as the dataset did not contain any pre-augmented images.

The second filtering operation dealt with duplicates in the datasets as they increase the risk of overfitting. Duplicates were removed based on a method proposed by Li et al. (2016) which leverages the idea of perceptual hash values to assess the similarity between the images in each dataset.This method aligns well with the described duplicate-removal approach in the dataset preparation of Li et al. (2016). By leveraging perceptual hashing:

- It removes exact duplicates and near-identical images caused by minor variations (e.g., encoding differences).
- Ensures a cleaner dataset for classification tasks, improving model training reliability.

rotated_by_30_Screen Shot 2018-06-12 at 11.37.13 PM.png

rotated_by_30_Screen Shot 2018-06-12 at 11.37.31 PM.png

Figure 6: Duplicate Data from "oranges_rotten" folder

Finally, the data was checked for anomalies related to the aspect ratio and RGB colors. Checking the aspect ratio is relevant when it comes to resizing images to a size that can be processed by the modelling algorithms. To avoid images and their features becoming too stretched out when being rescaled during the data normalization phase, all images with an aspect ratio of 4:1 and a square root size less than 300 pixels were filtered out from the data. The second set of anomalies was related to RGB anomalies. Here, all images with an average value higher than 250 or less than 5 in any of the three color channels were filtered out. This requirement ultimately resulted in removing images that were too bright or too dark on average. A complete overview of the filtering approach and how many images are removed in each step is shown in Figure 7.

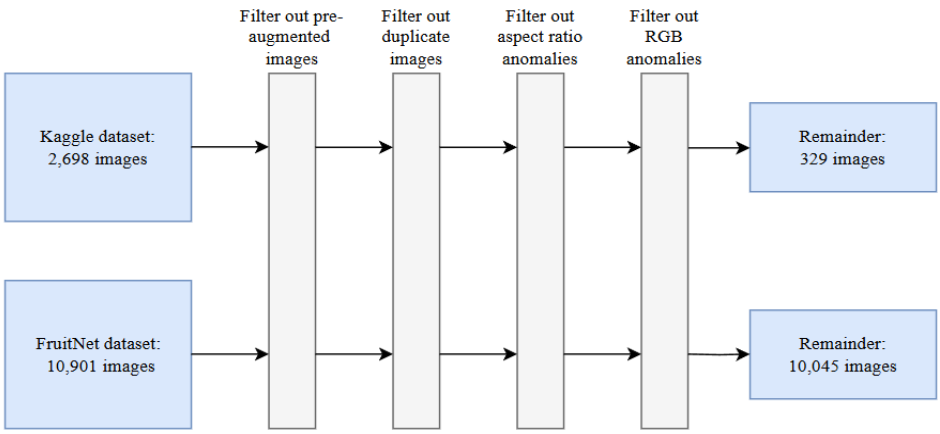


Figure 7: Data Filtering Process

4.2.2 Data Normalization

Next, the data needed to be normalized. This is a crucial step to achieve consistency throughout and across the datasets so that learning algorithms are able to work properly. The two main tasks in this stage deal were resizing and rescaling all images. As observed in Chapter 4.1, the images showed a great variation in their square root size and aspect ratio. This could have led to problems when attempting to train some classifiers as they often expect aligned formats across all data points. Thus, each image was resized to a size of 128x128 pixels which aimed to achieve a compromise between a feasible Runtime of the training algorithm and a sufficient level of detail for the algorithm to differentiate between the two classes. In the second normalization step, all pixels got rescaled to a value range between 0 and 1 to guarantee a consistent scale across all data points. I omitted a conversion to grayscale on purpose as I assessed the importance of the original colors as high in the context of identifying rotten spots on fruit. After adjusting the color scales, the datasets were ready to be merged, shuffled and split into a train, test and validation set using a 70:10:20 split.
Furthermore, having aligned data allowed for performing a Principal Component Analysis (PCA). The difficulty here was that data was represented by a four-dimensional array while a PCA is usually designed to handle two-dimensional data only. In order to use it despite this issue, I needed to split each image into its RGB color channels and then apply the PCA to all channels of each image individually. This ultimately led to varying principal components for different images which ultimately made me decide against including the PCA in the Preprocessing process. Nevertheless, I kept the PCA for exploratory purposes and plotted some of the reconstructed images which can be found in Appendix 4.

4.2.3 Data Augmentation

The last phase of the Preprocessing process was augmenting the data. This refers to the operation of applying different kind of changes or noise to the original data to obtain transformed copies of the same images. By this, I intended to increase the robustness of the model as well as the ability to generalize. Additionally, it provides aid in situations where there is not enough data available to train algorithms on. For the sake of this project, each image was augmented once which led to all data splits being doubled in size. This was done by utilizing Keras *ImageDataGenerator* class which provides a host of different augmentation techniques. The main benefit of using the tool was to provide real-time Data Augmentation so that random transformations are applied on each training image as it is passed to the model. That way, it requires lower memory usage than it would for doing the augmentation manually. From the wide range of different augmentation techniques implemented in the *ImageDataGenerator* class, I chose the following options: Rotations, vertical and horizontal shifts, shears, adjustments in the brightness and horizontal as well as vertical flips of the pictures. The options that were applied were chosen randomly by the *ImageDataGenerator* for each image. The

same applies for the intensity to which the transformation was applied. The final size of each dataset pre and post Data Augmentation is illustrated in Figure 8.

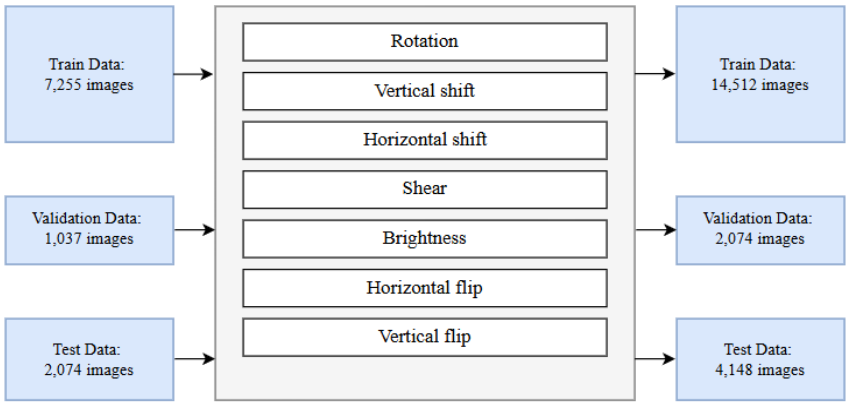


Figure 8: Data Augmentation

4.3 Modelling

4.3.1 Convolutional Neural Network

Convolutional Neural Networks belong to the most popular classes of Neural Networks. Since CNN achieved exceptional results on the *ImageNet Large Scale Visual Recognition Competiton*, they have become the dominant method in computer vision tasks (Krizhevsky et al., 2012[10]). CNN usually consist of multiple layers of different layer classes. The most frequently used layer classes in CNN are: Convolutional layers (1), pooling layers (2), flattening layers (3), dense layers (fully connected layers) (4), dropout layers (5), batch normalization layers (6). Convolutional layers are at the core of every CNN. They perform feature extraction via convolution: A kernel, which is a small matrix of numbers (e.g. 3x3), hovers over the input matrix (tensor) and calculates an element-wise product (Yamashita et al., 2018[11]). The resulting products are mapped into a feature map, as illustrated in Figure 9.

10 ImageNet Classification with Deep Convolutional Neural Networks
11 Convolutional neural networks: an overview and application in radiology

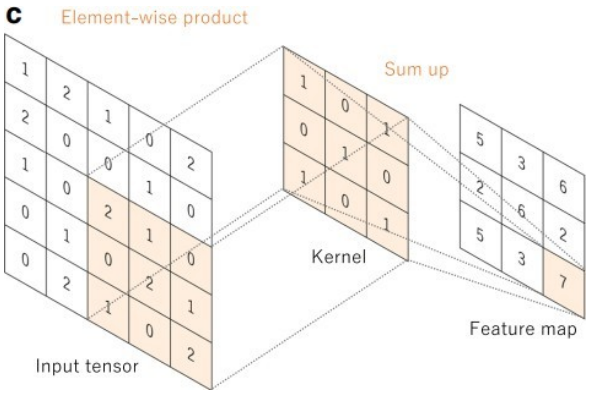


Figure 9: Example of Convolution Operation with a Kernel Size of 3 × 3 (Yamashita et al., 2018)

Pooling layers are often placed behind convolutional layers. They downsample the feature map to make pattern recognition more robust and less vulnerable to small variations, such as in the location of an object within the image. Flattening layers flatten the feature map into a single array. Dense layers, also called fully connected layers, are often applied at later stages of a CNN. They are called fully connected layers, since every neuron is connected to every neuron of the preceding layer. Dropout layers are often used to avoid overfitting of the model. They randomly shut off nodes during training, which is a computationally efficient and effective method to achieve this goal (Brownlee, 2018[12]). Finally, batch normalization layers can be used to stabilize and shorten the learning process via the application of standardization steps.

4.3.2 Transfer Learning

The transfer of knowledge and skills can be utilized not only by humans, but also by machines. Neural networks can apply knowledge that they have gained in one area to similar areas. This is known as Transfer Learning.

Convolutional Neural Networks are well suited for Transfer Learning. Python's open source Deep Learning library Keras offers multiple models that are applicable for this purpose. The models come with pre-trained weights and can be used for prediction, feature extraction and fine-tuning (Keras, 2022[13]). Transfer Learning can be approached with a three steps process (Xiang et al., 2019[14]): Select a base model (1), train the model (2) and fine-tune the model (3). A selected model (1) can be adjusted, for instance by adding further layers to it and adjusting the input layer to make it compatible with the respective input data. Regarding the training of the model (2), it is not only possible to train the entire model, but also to 'freeze' certain layers, which means the weights for the selected layers are not adjusted. In Transfer Learning, early layers are typically frozen while latter layers are trained

12 A Gentle Introduction to Dropout for Regularizing Deep Neural Networks
13 Keras Applications
14 Fruit Image Classification Based on MobileNetV2 with Transfer Learning Technique

on the new dataset. Finally, the model can be further optimized and adjusted to the specific task (3).

4.3.3 Applied Models

I have applied three different models on the rotten Fruit Classification task. First, two self-constructed CNN were applied. The first CNN (CNN 1) had a simplistic architecture consisting of eight different layers, the second CNN (CNN 2) has a higher variety of layers consisting of 19 layers in total. The two self-constructed CNN were compared to a pre-trained VGG16 model (in folder ”Final_Models”).

CNN1

The architecture of the first self-constructed CNN has been adapted from Mazumdar (2022). It consists of eight layers. The idea here was to train a simple CNN to get a first impression of the performance. It begins with two 2D convolutional layers followed by a pooling layer. Afterwards, a flattening layer and a dropout layer with 25% dropout are applied. Finally, the CNN ends with a dense layer, a dropout layer with 50% dropout and the final dense layer. The dropout layers reduce the overfitting of the model.

Training: CNN1 is initially trained on the training set, hence no Transfer Learning was applied.

CNN2

The architecture of CNN2 has been adapted from Kapse (2020)[15]. CNN2 is deeper than CNN 1. To avoid overfitting and to reduce the generalization error multiple Pooling, Normalization and Dropout layers are included in the model. It consists of 19 layers in total: Four convolutional layers, three pooling layers, one flattening layer, five batch normalization layers, four dropout layers and two dense layers. Training: CNN 2 is initially trained on the training set, hence no Transfer Learning was applied.

VGG16

Finally, I decided to include one pre-trained model as a benchmark for the performance of my self-constructed models. VGG16 is a CNN that was created by K. Simonyan and A. Zisserman from the University of Oxford (Simonyan & Zisserman, 2014[16]). The model was presented in the ImageNet Large Scale Visual Recognition Challenge 2014. The model achieved an Accuracy of 92.7% on the ImageNet dataset, which is a dataset that contains over 14 million images that are labeled into 1,000 classes. These classes also include several kinds of fruits and vegetables. ImageNet includes labels for bananas, but no labels for apples, limes and guavas.

As the name says, VGG16 consists of 16 layers that have weights (thirteen convolutional layers and three dense layer) and five pooling layers (Simonyan & Zisserman, 2014). I added a flattening layer, a

15 Fruit + quality classifier for NOOBS

16 Very Deep Convolutional Networks for Large-Scale Image Recognition

dense layer, a dropout layer and a final dense output layer to the VGG16. The dropout layer was added to avoid overfitting.
Training: I used the VGG16 with the pre-trained weights from the ImageNet dataset. I froze the weights of the original model and trained only the layers that were added by me.

An overview of the architecture of each model can be found in Appendix 6 for CNN1, Appendix 7 for CNN2 and Appendix 8 for VGG16.

5 Results

5.1 Performance Overview

I evaluated the performance of the three models based on the following metrics: Accuracy, Recall, Recall and F1-Sore. A comparison of the model across the different metrics is shown in Figure 10

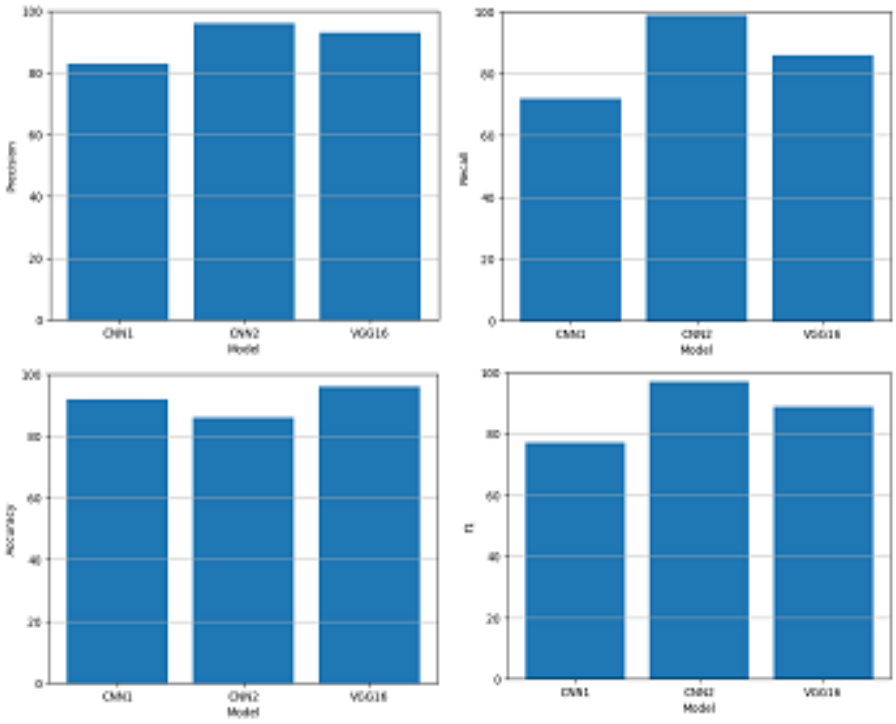


Figure 10: Performance Metrics of Applied CNN

Summary Table

Model	Accuracy	Loss	Fresh F1	Rotten F1	Insights
CNN1	0.9239	0.1931	0.74	0.77	High accuracy, but imbalanced precision/recall for both classes.
CNN2	0.8640	0.5231	**0.96**	**0.97**	Best for balanced detection, excels at detecting rotten fruit.
VGG16	**0.9674**	**0.0946**	0.87	0.89	Best accuracy and lowest loss, good overall performance.

Model Accuracy Overview:

- **CNN1**: **Accuracy = 0.9239**
  - CNN1 achieves a very good accuracy (92.39%) but struggles with imbalanced performance between the two classes, particularly with lower precision for fresh fruit and recall for rotten fruit.
- **CNN2**: **Accuracy = 0.8640**
  - CNN2 has a slightly lower accuracy (86.40%) than CNN1, but its **classification metrics** are well-balanced and provide excellent detection for both classes. Particularly, it excels in detecting rotten fruit with a recall of **0.99**.
- **VGG16**: **Accuracy = 0.9674**
  - VGG16 achieves the highest accuracy (96.74%) with the lowest loss (0.0946). It performs well for both classes, though its recall for rotten fruit is slightly lower (**0.86**) compared to CNN2.

Model Selection Based on Accuracy and Performance:

- **Best Accuracy**: **VGG16** stands out with the **highest accuracy** of 96.74% and the **lowest loss**, making it the most efficient model in terms of performance.
  - **Strengths**: High overall accuracy, good precision and recall, particularly for fresh fruit with a precision of **0.93**.
  - **Weakness**: Slightly lower recall for rotten fruit (**0.86**) compared to CNN2, but it still remains effective.
- **Best for Balanced Performance**: **CNN2**, despite the lower accuracy (86.40%), provides the **best balance** in detecting both fresh and rotten fruit, with particularly high recall for rotten fruit (**0.99**).
  - **Strengths**: Reliable detection of rotten fruit, balanced performance for both classes, precision and recall are both strong across the board.
  - **Weakness**: Slightly lower accuracy than VGG16 and CNN1, but with a strong focus on detection, particularly for rotten fruit.
- **Balanced but Imbalanced Precision/Recall**: **CNN1** has solid performance but is marked by an imbalance. It has a high precision for rotten fruit (**0.83**) but a much lower recall (**0.72**), leading to missed detections for rotten fruit. It has **good accuracy** (92.39%), but the class imbalance impacts its overall effectiveness.

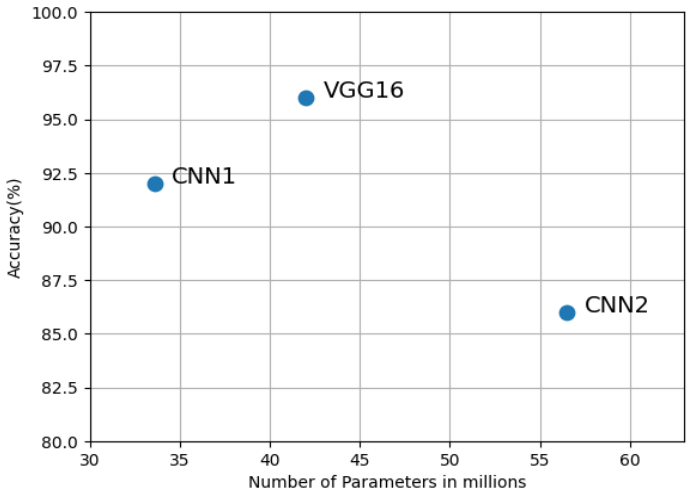


Figure 11: Accuracy and Parameters of Applied CNN

5.2 Runtime Analysis

The analysis of the Runtime is a crucial step in the evaluation of a model. A model with a lower Accuracy can be preferred if the Runtime is faster. Figure 12 shows a visualization of the Runtimes of the different models.

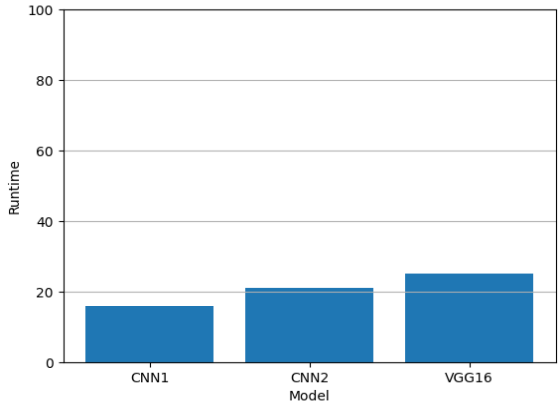


Figure 12: Runtime Comparison

All models have been trained on Google Colab Pro+ on a machine with 50 GB of RAM and with the help of a Tensor Processing Unit (TPU) to speed up the training process. A TPU is a processor unit specifically designed to speed up Machine Learning algorithms based on tensors.

- VGG16 Model (1534 sec or 25 minutes):
    - Even though VGG16 is generally a more complex model with a deep architecture, its pretrained layers are frozen (weights not updated), which means only the final layers are

trained. This significantly reduces the computational burden.

- Faster runtime is a result of leveraging the pretrained weights, as no training occurs in the initial convolutional layers.

- CNN1 Model (1000 sec or 16 minutes):
  - The longer runtime for CNN1, despite having fewer layers than CNN2, is due to the higher number of trainable parameters and no pretrained weights being used. This means the model is fully trained from scratch, which can take longer depending on the size and complexity of the architecture.
  - The model trains all layers, including the convolutional layers, leading to longer training times, even with fewer layers overall.

- CNN2 Model (1254 sec or 21 minutes):
  - Even though CNN2 has more layers, it still performs faster than CNN1 due to multiple pooling and normalization stages that reduce the number of trainable parameters.
  - The lower number of trainable parameters helps accelerate the training process, despite the model being more complex in terms of layers.

Key Takeaways:

- VGG16: The benefit of using a pretrained model (with frozen layers) leads to the fastest runtime. It only trains the added layers, making it highly efficient despite its deep architecture.
- CNN1: This model takes the longest time to train because it has more trainable parameters and no pretrained weights, leading to slower convergence and higher computational demands.
- CNN2: The model performs faster than CNN1 due to pooling and normalization layers that reduce the number of trainable parameters, even though it has more layers.

5.3 Limitations

There are several limitations to the approach presented in this project. First of all, the Classification only regarded whether some fruit was fresh or rotten without taking the type of fruit into account. Such extension would be relevant for e.g. providers like supermarkets in order to analyze which types of fruit might be more prone to rot and therefore might need a more careful treatment when it comes to storing and packaging. Additionally, having a model that is able to identify different fruits could also be leveraged for increasing the performance of the classifier e.g., by using it to identify single fruit instances in an image which contains multiple instances at once.

A second issue was the limited variation of different types of fruit in the data. Even though the models perform very well on the three given types of fruit, I can't expect it to generalize and work properly on unseen types too. Nevertheless, the model can be seen as a starting point which could be extended through the utilization of Transfer Learning. Furthermore other types of classifiers and the comparison to non neural networks would have led to better benchmarking of the models.

One last limitation which was already touched upon in Chapter 4.2.2 was the deliberate decision against using a Grayscale. Even though I assumed that the relevance of using RGB colors outweighed the computation advantages of using Grayscale image data in the context of this project, it might still be an interesting subject for future research to examine the actual performance loss when using Grayscale data instead of RGB data.

6 Conclusion

The goal of this project was to set up a classifier that could accurately predict whether a given fruit is rotten or not. Therefore, I built two different CNN and compared their performances with a pre-trained VGG16 model as a benchmark. All three models performed well in terms of Accuracy, Recall, Recall and F1-score. As anticipated, the benchmark model reached the highest performance values on average across all measures. And although the Classification task can also be managed fairly by a self- constructed deep CNN, it would still not pay off to choose it over the VGG16 from a Complexity i.e. Runtime perspective. Finally, the results indicate that it is possible to predict the quality of fruits with CNN. Therefore, the project can be seen as a starting point for the implementation of automated recognition of rotten fruits, a capability that will become increasingly relevant on a way towards a more sustainable food value chain.

References

Bhargava, A., & Bansal, A. (2021). Fruits and vegetables quality evaluation using computer vision: A review. *Journal of King Saud University - Computer and Information Sciences*, *33*(3), 243–257. https://doi.org/10.1016/j.jksuci.2018.06.002

Brownlee, J. (2018, December 2). A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. *Machine Learning Mastery*. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Dandavate, R., & Patodkar, V. (2020). CNN and Data Augmentation Based Fruit Classification Model. *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 784–787. https://doi.org/10.1109/I-SMAC49090.2020.9243440

Food wastage footprint: Impacts on natural resources: summary report. (2013). FAO.

Food Waste, by the Numbers. (2018). https://foodforward.org/food-waste/food-waste-numbers/

Hossain, M. S., Al-Hammadi, M., & Muhammad, G. (2019). Automatic Fruit Classification Using Deep Learning for Industrial Applications. *IEEE Transactions on Industrial Informatics*, *15*(2), 1027–1034. https://doi.org/10.1109/TII.2018.2875149

Kapse, A. (2020). *Fruit classifier—Kaggle*. https://www.kaggle.com/code/akhileshdkapse/fruit-quality-classifier-for-noobs

Kausar, A., Sharif, M., Park, J., & Shin, D. R. (2018). Pure-CNN: A Framework for Fruit Images Classification. *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 404–408. https://doi.org/10.1109/CSCI46756.2018.00082

Keras. (2022). *Keras documentation: Keras Applications*. https://keras.io/api/applications/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

Mazumdar, S. (2022). *CBS Exercise: Data Mining and Machine Learning*.

Meshram, V., & Patil, K. (2022). FruitNet: Indian fruits image dataset with quality for machine learning applications. *Data in Brief*, *40*, 107686. https://doi.org/10.1016/j.dib.2021.107686

Siddiqi, R. (2019). Effectiveness of Transfer Learning and Fine Tuning in Automated Fruit Image Classification. *Proceedings of the 2019 3rd International Conference on Deep Learning Technologies*, 91–100. https://doi.org/10.1145/3342999.3343002

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. https://arxiv.org/abs/1409.1556

Xiang, Q., Wang, X., Li, R., Zhang, G., Lai, J., & Hu, Q. (2019). Fruit Image Classification Based on MobileNetV2 with Transfer Learning Technique. *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 1–7. https://doi.org/10.1145/3331453.3361658

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, 9(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Zhang, Y., Wang, S., Ji, G., & Phillips, P. (2014). Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, *143*, 167–177. https://doi.org/10.1016/j.jfoodeng.2014.07.001

Zhang, Y.-D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K., & Wang, S.-H. (2019). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimedia Tools and Applications*, *78*(3), 3613–3632. https://doi.org/10.1007/s11042-017-5243-3

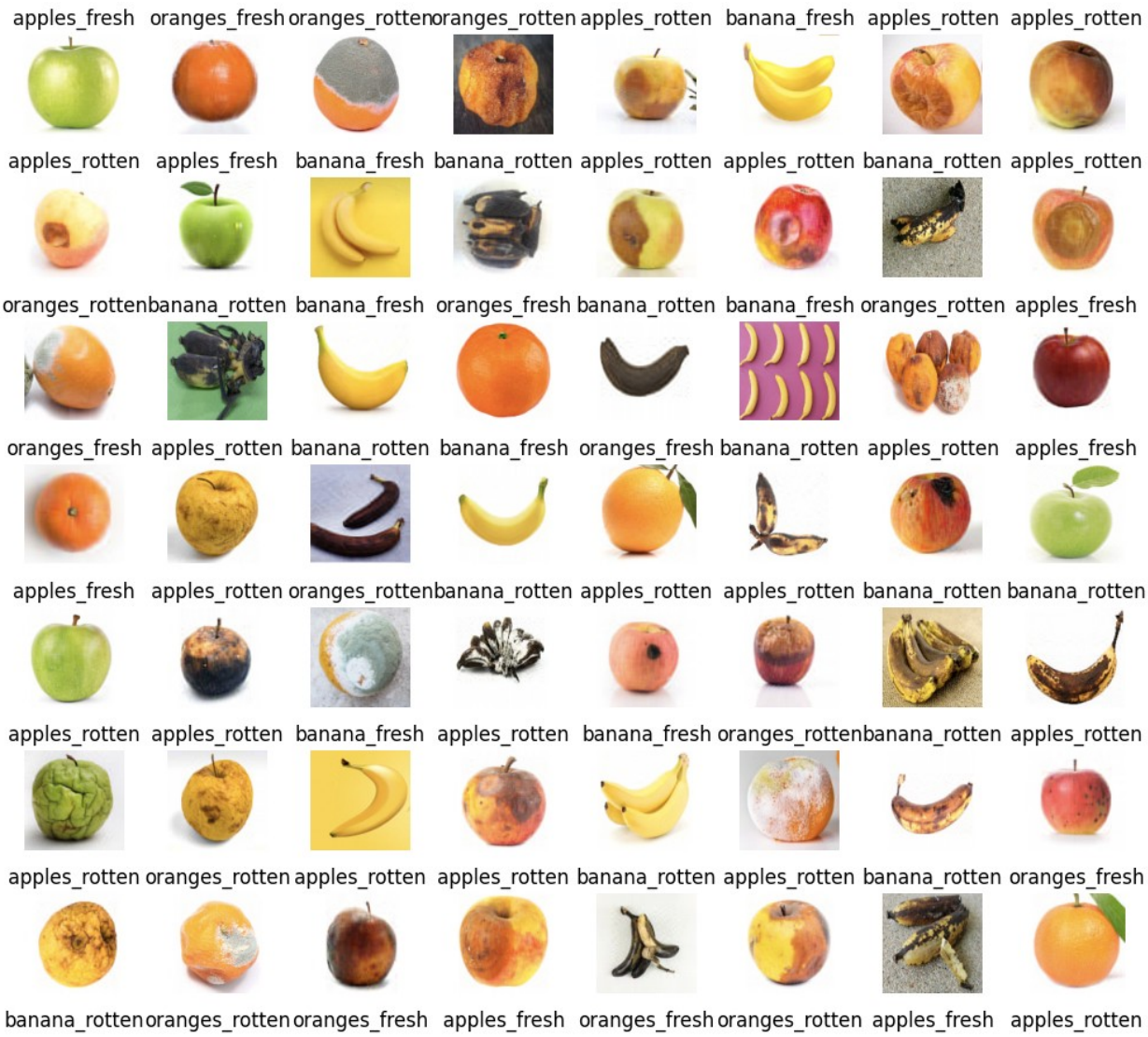
Appendix

Appendix 1: Image examples of Kaggle dataset

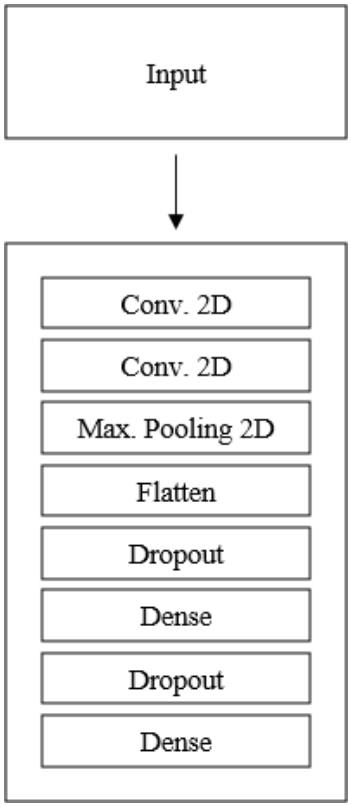
Appendix 2: Image examples of FruitNet dataset

Dataset	Quality	Apple	Banana	Orange	Total
Kaggle	Fresh	395 (14.64%)	381 (14.12%)	388 (14.38%)	1164 (43.14%)
	Rotten	601 (22.28%)	530 (19.64%)	403 (14.94%)	1534 (56.85%)
FruitNet	Fresh	1693 (15.53%)	1581 (14.50%)	1466 (13.45%)	4740 (43.43%)
	Rotten	2342 (21.48%)	2224 (20.40%)	1595 (14.63%)	6161 (56.51%)

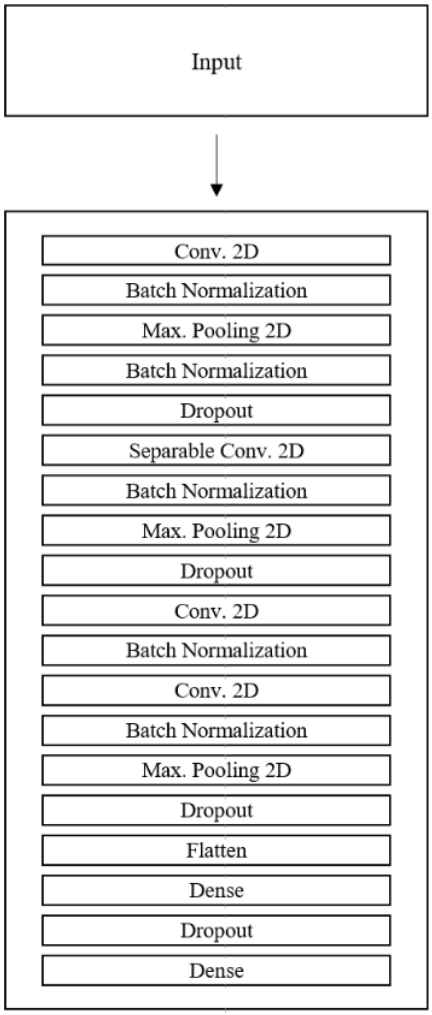
Appendix 3: Distribution of fruits and quality classes among datasets



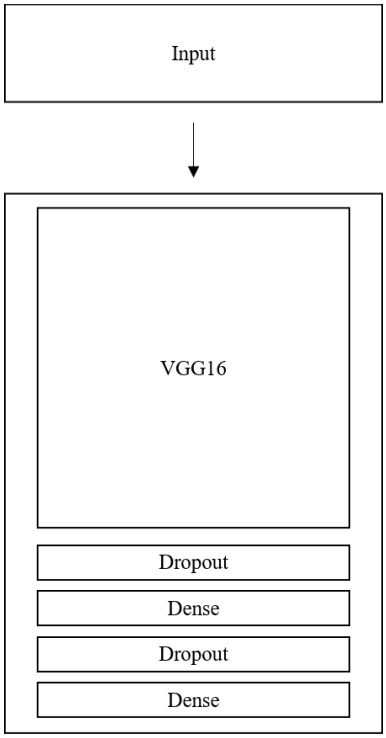
Appendix 4: Reconstructed images from PCA



Appendix 6: Model architecture of CNN1



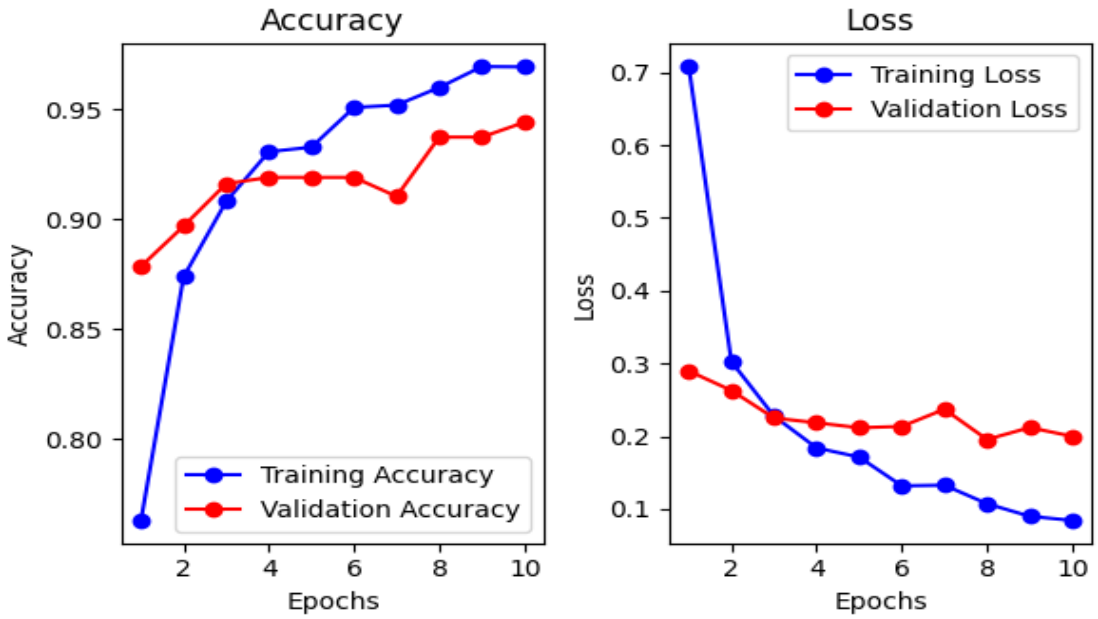
Appendix 7: Model architecture of CNN2



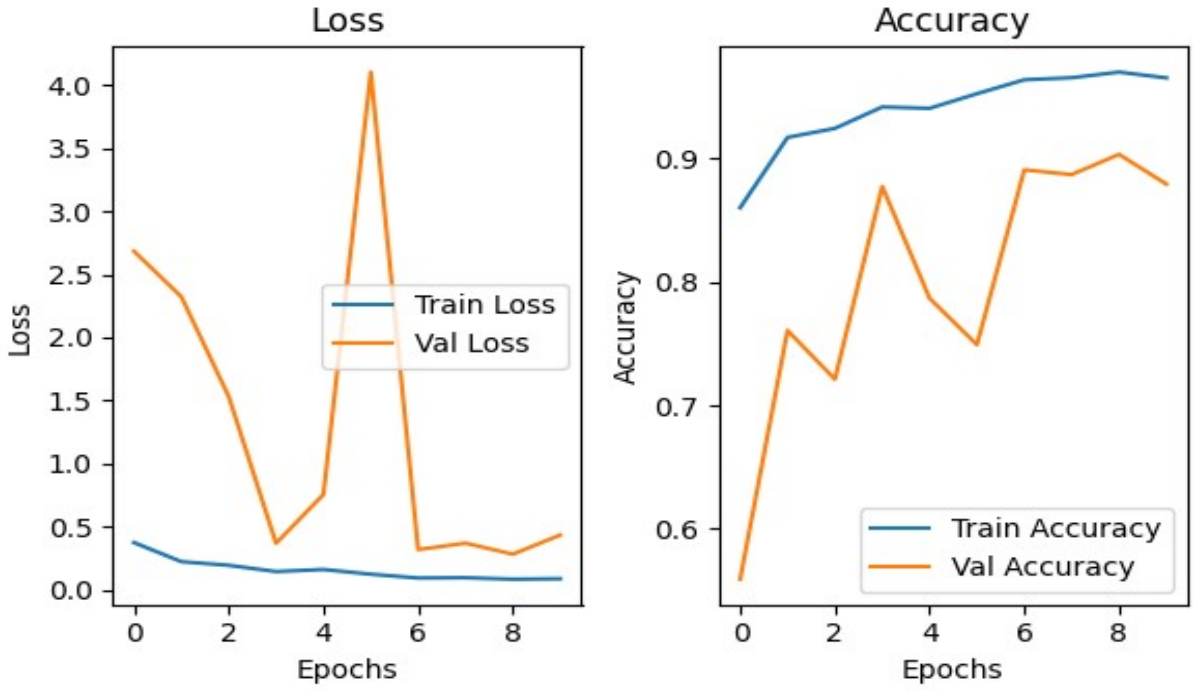
Appendix 8: Model architecture of VGG16

bv

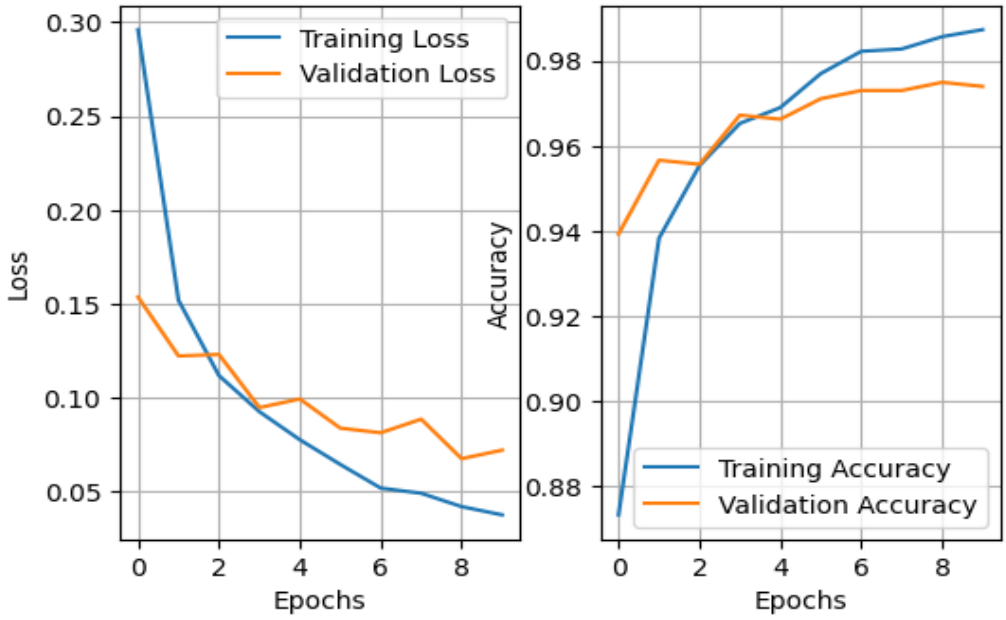
Appendix 9: Examples of wrongly predicted fruits of the CNN2 model



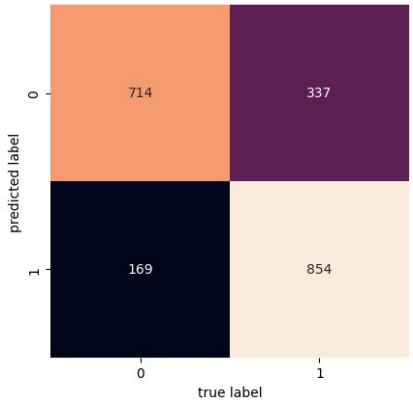
Appendix 10: CNN1 Training History



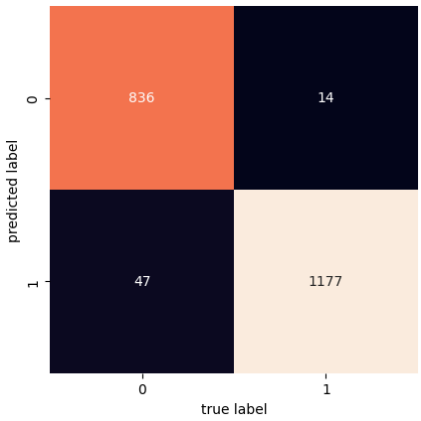
Appendix 11: CNN2 Training History



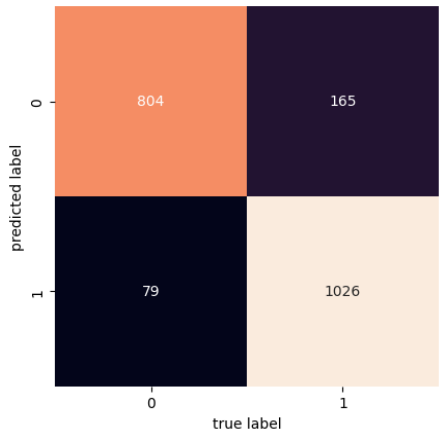
Appendix 12: VGG16 Training History



Appendix 13: Confusion Matrix CNN1



Appendix 14: Confusion Matrix CNN2



Appendix 15: Confusion Matrix VGG16